



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
-----------------	-------------	----------------------	---------------------	------------------

10/706,515

11/12/2003

Fei Luo

BEAS-1339US2

7689

23910 7590 08/09/2007
FLIESLER MEYER LLP
650 CALIFORNIA STREET
14TH FLOOR
SAN FRANCISCO, CA 94108

EXAMINER

ZHEN, LI B

ART UNIT

PAPER NUMBER

2194

MAIL DATE

DELIVERY MODE

08/09/2007

PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary

Application No.

10/706,515

Applicant(s)

LUO ET AL.

Examiner

Li B. Zhen

Art Unit

2194

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) ☒ Responsive to communication(s) filed on 04 June 2007.
- 2a) ☐ This action is **FINAL**. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) ☒ Claim(s) 4-6,9,10,13,16 and 24-27 is/are pending in the application.
- 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) ☐ Claim(s) _____ is/are allowed.
- 6) ☒ Claim(s) 4-6,9,10,13,16 and 24-27 is/are rejected.
- 7) ☐ Claim(s) _____ is/are objected to.
- 8) ☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☐ The drawing(s) filed on _____ is/are: a) ☐ accepted or b) ☐ objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some * c) ☐ None of:
- ☐ Certified copies of the priority documents have been received.
 - ☐ Certified copies of the priority documents have been received in Application No. _____.
 - ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- 1) ☒ Notice of References Cited (PTO-892)
- 2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
- 3) ☒ Information Disclosure Statement(s) (PTO/SB/08)
Paper No(s)/Mail Date 06/04/2007.
- 4) ☐ Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____
- 5) ☐ Notice of Informal Patent Application
- 6) ☐ Other: _____

DETAILED ACTION

1. Claims 4 – 6, 9, 10, 13, 16 and 24 – 27 are pending in the application.

Response to Arguments

2. Applicant's arguments with respect to the claims have been considered but are moot in view of the new ground(s) of rejection.

Claim Rejections - 35 USC § 103

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

4. This application currently names joint inventors. In considering patentability of the claims under 35 U.S.C. 103(a), the examiner presumes that the subject matter of the various claims was commonly owned at the time any inventions covered therein were made absent any evidence to the contrary. Applicant is advised of the obligation under 37 CFR 1.56 to point out the inventor and invention dates of each claim that was not commonly owned at the time a later invention was made in order for the examiner to consider the applicability of 35 U.S.C. 103(c) and potential 35 U.S.C. 102(e), (f) or (g) prior art under 35 U.S.C. 103(a).

5. Claims 4 – 6, 9, 26 and 27 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 6,993,774 to Glass [cited in the previous office action] in view of U.S. Patent No. 7,146,399 to Fox et al. [hereinafter Fox].

6. As to claim 26, Glass teaches the invention substantially as claimed including a computer-readable medium carrying instructions for dynamically generating a wrapper object [dynamic generation of remote proxies; col. 6, lines 40 – 55], comprising the steps of:

receiving a resource adapter class [reads the associated class 252 from a class repository, col. 18, lines 56 – 63, see Fig. 11, element 252 can be either class or object; Glass also discloses locating the subject object, step 26, Fig. 2, col. 7, lines 19 - 35; Examiner notes that the specification does not specifically define a vendor object, therefore a vendor object is given its plain meaning and is interpreted as object that provides services to other applications. The subject object as disclosed in Glass exists on a server system and provides services to clients, see col. 8, lines 1 – 12. Therefore, the subject object as disclosed in Glass corresponds to the recited vendor object.] at an application server [server systems 12; col. 4, line 62 – col. 5, line 8];

performing reflection on the resource adapter class [invokes reflection engine 36 to determine information regarding subject class 19; col. 8, lines 1 – 12] to identify interfaces implemented by the resource adapter class [proxy object 22 which contains the interfaces; col. 6, lines 40 – 55];

dynamically generating a wrapper class at runtime [generate the byte codes that define the class of subject object 18, col. 6, line 55 – col. 7, line 6; remote proxy for the subject object will inherit all of the variables and methods of its ancestors; col. 7, lines 58 – 67];

instantiating a wrapper object from the wrapper class [class loader 46 takes the generated bytes of remote proxy class 23 stored in memory and loads them into a class structure which then can be instantiated to create remote proxy object 22; col. 10, lines 1 – 10]; and

providing the wrapper object [generated interface is associated with subject class 19; col. 8, lines 40 – 48] to an application that requires support for the interfaces implemented by the resource adapter class [col. 6, lines 40 – 55].

Although Glass teaches the wrapper class implements the interfaces identified through reflection [invokes reflection engine 36 to determine information regarding subject class 19; col. 8, lines 1 – 12], Glass does not specifically disclose generating a wrapper class that extends from a superclass, wherein the superclass implements Java Database Connectivity, Java Messaging Service, or Java Connector Architecture interfaces.

However, Fox teaches generating a wrapper class [vendors of EAI systems recommend use of an adapter; col. 14, lines 37 – 46] that extends from a superclass [implement either (i) proprietary application programming interfaces (APIs) exposed by Message Broker 820, such as Tibco Message Broker or WebsphereMQ Integrator, or (ii) cross-platform APIs, such as Java Connector Architecture; col. 14, lines 37 – 46],

Art Unit: 2194

wherein the superclass implements Java Database Connectivity, Java Messaging Service, or Java Connector Architecture interfaces [Java Connector Architecture; col. 14, lines 37 – 46].

It would have been obvious to a person of ordinary skill in the art at the time the invention was made to modify the invention of Glass to incorporate the features of a wrapper class that extends from a superclass, wherein the superclass implements Java Database Connectivity, Java Messaging Service, or Java Connector Architecture interfaces because this provides communication with outside software [col. 14, lines 27 – 36 of Fox] and allows data to be transformed from one schema to another [col. 5, lines 2 – 15 of Fox].

7. As to claim 4, Glass teaches the superclass includes logic to handle server side tasks [forwards the message to the appropriate EJB function object 206 for preliminary processing; col. 15, lines 38 – 56].

8. As to claim 5, Glass teaches the wrapper class is generated in bytecode [byte codes representing remote proxy class 23 are generated; col. 7, lines 20 – 35].

9. As to claim 6, Glass teaches bytecode is generated for vendor methods [byte codes representing remote proxy class 23 are generated; col. 7, lines 20 – 35] not implemented in the superclass [superclass remote proxies; col. 7, lines 56 – 67; examiner notes that the superclass remote proxies include all of the variables and

methods of the subject class' ancestors, therefore, the subject class would include methods that are not implemented in the superclass].

10. As to claim 9, Glass teaches Enterprise Java Beans [col. 15, lines 1 – 16] and Enterprise JavaBeans technology is the server-side component architecture for Java Platform, Enterprise Edition (Java EE).

11. As to claim 27, Glass as modified teaches the superclass is statically predefined [Java Connector Architecture; col. 14, lines 37 – 46 of Fox].

12. Claims 10, 13, 16, 24 and 25 are rejected under 35 U.S.C. 103(a) as being unpatentable over Glass and Fox further in view of U.S. Patent Application Publication No. 2004/0143835 to Dattke et al. [hereinafter Dattke, cited in the previous office action].

13. As to claim 10, Glass as modified teaches the invention substantially as claimed including a computer-readable medium carrying instructions for processing an invocation at a dynamically generated wrapper [dynamic generation of remote proxies; col. 6, lines 40 – 55 of Glass], comprising the steps of:

receiving, from an application [Local object 20 may request access to subject object 18; col. 5, line 52 – col. 6, line 7 of Glass], an invocation by a wrapper object, the wrapper object instantiated from a wrapper class [vendors of EAI systems recommend

use of an adapter; col. 14, lines 37 – 46 of Fox], the wrapper class extended from a superclass [implement either (i) proprietary application programming interfaces (APIs) exposed by Message Broker 820, such as Tibco Message Broker or WebsphereMQ Integrator, or (ii) cross-platform APIs, such as Java Connector Architecture; col. 14, lines 37 – 46 of Fox] which implements the Java Database Connectivity, Java Messaging Service, or Java Connector Architecture interfaces [Java Connector Architecture; col. 14, lines 37 – 46 of Fox], the invocation [In order to isolate the distributed processing communication requirements from local object 20, a remote proxy object 22 may be created on server system 12 and loaded onto client system 14; col. 5, line 52 – col. 6, line 7 of Glass] directed to a wrapped resource adapter [proxy object 22 which contains the interfaces; col. 6, lines 40 – 55 of Glass];

initiating pre-processing by calling a pre-invocation handler configured to execute server-side code [Type object 204 forwards the message to the appropriate EJB function object 206 for preliminary processing; col. 15, lines 38 – 56 of Glass];

calling the wrapped object [Local object 20 communicates with remote proxy object 22 which then communicates with subject object 18; col. 5, line 52 – col. 6, line 7 of Glass];

receiving a result from the wrapped object [Reference object 158 decodes the result and passes it to remote proxy 154; col. 13, lines 40 – 58 of Glass];

initiating post-processing by calling a post-invocation handler configured to execute post processing server-side tasks [Set of streamers 180 handles the encoding

and transmission of arguments and results according to the communication protocol used by the receiving object; col. 14, lines 13 – 31 of Glass]; and

providing the result to the application [Remote proxy 154 then makes the result available to client application 108; col. 13, lines 40 – 58 of Glass] thereby enabling the application to access vendor specific methods [Communications between client application 108 and server object 110 proceed by client application 108 communicating with remote proxy 154 through its interface IProxy 152; col. 13, lines 25 – 40 of Glass] of the wrapped resource adapter [proxy object 22 which contains the interfaces; col. 6, lines 40 – 55 of Glass]. Although Glass as modified teaches the invention substantially, Glass as modified does not specifically teach enabling the application program to access vendor specific extension methods of the wrapped resource adapter.

However, Dattke teaches an application extension runtime environment for vendor objects [an application extension runtime environment 100 for standard applications; pp. 2-3, paragraph 0031], generating a wrapper class [dynamic proxy] comprising at least one of vendor specific extension methods [extension object] from the vendor [standard application; p. 4, paragraph 0038] class [generate a dynamic proxy for the an extension object implemented by the application extension; pp. 2-3, paragraph 0031], generating a wrapper object as an instance of the wrapper class by instantiating the wrapper class [p. 3, paragraph 0035], and enabling the application program to access vendor specific extension methods of the wrapped resource adapter [dynamic proxy for the extension object interface can also include interfaces to call the concrete methods of the extension object; p. 3, paragraph 0034].

It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to further modify the invention of Glass and Fox to incorporate the features of enabling the application program to access vendor specific extension methods of the wrapped resource adapter because this allows dynamic proxy classes to provide extensions to standard applications [p. 1, paragraph 0005 of Dattke] and permits standard applications to be modified as part of a future release without requiring any specific modifications to support existing application extension [p. 2, paragraph 0023 of Dattke]. Dynamic proxy classes can be used to create a type-safe proxy object for a list of interfaces without requiring pre-generation of the class prior to compilation [p. 1, paragraph 0005 of Dattke] and provides the advantage of allowing customers of the standard application to customize the features of the standard application by providing customer-specific extensions for the features implemented by the standard application [p. 1, paragraph 0002 of Dattke].

14. As to claim 24, Glass as modified teaches the invention substantially as claimed including a computer-readable medium carrying instructions for processing an invocation at a dynamically generated wrapper [dynamic generation of remote proxies; col. 6, lines 40 – 55 of Glass], comprising the steps of:

receiving, from an application [Local object 20 may request access to subject object 18; col. 5, line 52 – col. 6, line 7 of Glass], a method invocation [In order to isolate the distributed processing communication requirements from local object 20, a remote proxy object 22 may be created on server system 12 and loaded onto client system 14;

col. 5, line 52 – col. 6, line 7 of Glass] to a resource adapter [proxy object 22 which contains the interfaces; col. 6, lines 40 – 55 of Glass];

calling a wrapper object for processing the method invocation [Local object 20 communicates with remote proxy object 22 which then communicates with subject object 18; col. 5, line 52 – col. 6, line 7 of Glass] wherein the wrapper object is dynamically generated [byte code generator 42 is to directly generate the executable code corresponding to JClass information 38. JClass information 38 is the definition of the Java class of which remote proxy object 22 is an instance; col. 9, lines 10 – 28 of Glass] from a resource adapter class [col. 14, lines 37 – 46 of Fox];

initiating pre-processing by the wrapper object, wherein the wrapper object calls a pre-invocation handler configured to perform server side logic [Type object 204 forwards the message to the appropriate EJB function object 206 for preliminary processing; col. 15, lines 38 – 56 of Glass];

forwarding the method invocation to the resource adapter [proxy object 22 which contains the interfaces; col. 6, lines 40 – 55 of Glass] by the wrapper object on behalf of the application [Local object 20 communicates with remote proxy object 22 which then communicates with subject object 18; col. 5, line 52 – col. 6, line 7 of Glass];

receiving a result of the method invocation from the resource adapter [proxy object 22 which contains the interfaces; col. 6, lines 40 – 55 of Glass] by the wrapper object [Reference object 158 decodes the result and passes it to remote proxy 154; col. 13, lines 40 – 58 of Glass];

initiating post-processing by the wrapper object, wherein the wrapper object calls a post-invocation handler configured to perform server-side logic [Set of streamers 180 handles the encoding and transmission of arguments and results according to the communication protocol used by the receiving object; col. 14, lines 13 – 31 of Glass]; and

providing the result to the application [Remote proxy 154 then makes the result available to client application 108; col. 13, lines 40 – 58 of Glass], thereby enabling the application to access vendor specific methods [Communications between client application 108 and server object 110 proceed by client application 108 communicating with remote proxy 154 through its interface IProxy 152; col. 13, lines 25 – 40 of Glass] of the resource adapter [proxy object 22 which contains the interfaces; col. 6, lines 40 – 55 of Glass]. Although Glass as modified teaches the invention substantially, Glass as modified does not specifically teach enabling the application program to access vendor specific extension methods of the wrapped vendor object.

However, Dattke teaches an application extension runtime environment for vendor objects [an application extension runtime environment 100 for standard applications; pp. 2-3, paragraph 0031], generating a wrapper class [dynamic proxy] comprising at least one of vendor specific extension methods [extension object] from the vendor [standard application; p. 4, paragraph 0038] class [generate a dynamic proxy for the an extension object implemented by the application extension; pp. 2-3, paragraph 0031], generating a wrapper object as an instance of the wrapper class by instantiating the wrapper class [p. 3, paragraph 0035], and enabling the application

program to access vendor specific extension methods of the wrapped resource adapter [dynamic proxy for the extension object interface can also include interfaces to call the concrete methods of the extension object; p. 3, paragraph 0034].

It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to modify the invention of Glass and Fox to incorporate the features of enabling the application program to access vendor specific extension methods of the wrapped resource adapter because this allows dynamic proxy classes to provide extensions to standard applications [p. 1, paragraph 0005 of Dattke] and permits standard applications to be modified as part of a future release without requiring any specific modifications to support existing application extension [p. 2, paragraph 0023 of Dattke]. Dynamic proxy classes can be used to create a type-safe proxy object for a list of interfaces without requiring pre-generation of the class prior to compilation [p. 1, paragraph 0005 of Dattke] and provides the advantage of allowing customers of the standard application to customize the features of the standard application by providing customer-specific extensions for the features implemented by the standard application [p. 1, paragraph 0002 of Dattke].

15. As to claim 13, Glass teaches the server-side code executed by the pre-invocation handler includes transaction processing code [Preliminary common processing may include...transaction management; col. 15, lines 38 – 57].

Art Unit: 2194

16. As to claim 16, Glass teaches the post-processing server-side tasks include transaction management [generated class functionality may include...transaction management; col. 15, lines 1 – 16].

17. As to claim 25, Glass teaches the server-side logic includes at least one of transaction management [generated class functionality may include...transaction management; col. 15, lines 1 – 16], pooling, caching, tracing and profiling.

CONTACT INFORMATION

18. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Li B. Zhen whose telephone number is (571) 272-3768. The examiner can normally be reached on Mon - Fri, 8:30am - 5pm.


If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, William Thomson can be reached on 571-272-3718. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Art Unit: 2194

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

Li B. Zhen
Examiner
Art Unit 2194

LBZ


8/5/2007